

REMARKS

The application was filed on 28 June 2001 with fifteen claims. The Examiner on 21 April 2004 issued a first Action. In the action, the Examiner objected to the drawings and issued several objections to the specification and the title. With respect to the claims, the Examiner objected to claim 5 and rejected claims 1-10 under 35 U.S.C. §112. The Examiner issued four art rejections of the claims: claims 1, 5, 10-12 were rejected under 35 U.S.C. §102(b) as being anticipated by U.S. Patent 6,629,271B1 entitled TECHNIQUE FOR SYNCHRONIZING FAULTS IN A PROCESSOR HAVING A REPLAY SYSTEM to Lee et al. (Lee '271); claims 2-4, 6 were rejected under 35 U.S.C. §103(a) as being unpatentable over Lee '271 in view of U.S. Patent No. 6,311,261 entitled APPARATUS AND METHOD FOR IMPROVING SUPERSCALAR PROCESSORS to Chamdani et al. (Chamdani '261) and U.S. Patent Application Publication 2002/0078317 A1 entitled FIRST-IN, FIRST-OUT (FIFO) MEMORY WITH MOVING BOUNDARY to Yasoshima (Yasoshima '317 Pub); claims 7-9 were rejected under 35 U.S.C. §103(a) as being unpatentable over Yasoshima '317 Pub in view of Lee '271; and claims 13-15 were rejected under 35 U.S.C. §103(a) as being unpatentable over Lee '271. In response, Applicants amended the title and the specification, submitted a proposed amendment to the drawing, amended independent claims 1, 6, 7, 10, 11, and 13 and dependent claims 5, 9, and 15, and cancelled claims 2 and 14.

The Examiner mailed a final rejection of the claims on 12 January 2005. Claims claims 1, 3-5 and 7-9 were finally rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent 6,353,829 B1 entitled METHOD AND SYSTEM FOR MEMORY ALLOCATION IN A MULTIPROCESSING ENVIRONMENT to Koblenz et al. (Koblenz '829). The Examiner rejected claim 5 under 35 U.S.C. §103(a) as being unpatentable over Koblenz '829 in view of Lee '271. The Examiner further rejected claims 6, 10-13,

and 15 under 35 U.S.C. §103(a) as being unpatentable over Lee '271 in view of Koblenz '829.

In response, Applicants insert the limitations of claim 3 into claim 1, cancel claim 3, and traverse the rejection of the combined claim and dependent claims. Applicants further amended claims 10 and 11. Applicants further traverse the rejections of claim 5, 6, 7-9, 12, 13, and 15 under the combination of Koblenz '829 and Lee '271. Applicants request the Examiner enter the amendments to put the application in condition for allowance or better condition for appeal. Claims 1, 4-13 and 15 are pending.

*The Rejection Under 35 U.S.C. §102(b) over Koblenz '829*

The Examiner rejected claims 1, 3-4, and 7-9 as being anticipated by Koblenz '829. In response, Applicants incorporate dependent claim 3 into independent claim 1 and assert that the amendment overcomes the rejection of claims 1 and 4. Applicants now assert that the claims are not anticipated by Koblenz '829 because Koblenz '829 does not teach or suggest a bank number to indicate how many times the head pointer has wrapped around the tail pointer in order to maintain an order of the resources for the at least one thread, as in independent claim 1; or incrementing a bank number if the first entry passes a last entry of the particular thread before it finds the free entry number as in independent claim 7.

Koblenz '829 teaches a memory allocation system in a multiprocessing system in which each processor may also be a multithreaded processor. Each processor in Koblenz '829 contains a complete set of registers for each stream (Koblenz '829, column 1, lines 23-24) and each thread of execution executes on one of 128 streams supported by a multithreaded architecture processor (column 1, lines 29-30). Every clock time period, the processor selects a stream that is ready to execute and allows it to issue its next instruction into a pipelined processor, the network, and the memory. Thus, a new instruction for a different stream may be

issued in each time period without interfering with other instructions that are in the pipeline. When an instruction finishes, the stream to which it belongs becomes ready to execute the next instruction. The state of a stream includes one stream status word (SSW), 32 general registers, eight target registers; thus, each processor has 128 sets of stream status words, 128 sets of general registers, and 128 sets of target registers.

Koblenz '829 teaches a data structure called a six list data structure that has a pointer to a circle portion having a number of items [warehouse header] currently in the circular list, and another pointer to a tail portion having a number of items [warehouse header] that have been removed from the circular list (column 11, lines 34-39). Each item [warehouse header] contains a next pointer and a last (or previous) pointer (column 13, lines 29-30), indicating the warehouse in memory to access data. It is unclear how many items are in the circular list - whether it is six, thirty-two, sixty-four; in any event, each item points to the next item, and in the embodiment shown, also has a pointer from the previous item.

The Examiner references Koblenz '829 at column 11, lines 10-55; column 14, line 54 to column 15, line 25, column 16, line 37 to column 17, line 6; Figure 4; Figure 11; and Figure 14. Attorney for Applicants have carefully read and reread and reread these referenced columns and line numbers and will now discuss each section thoroughly, in part to further understand that Koblenz '829 does not teach a "bank number" as claimed. After reading the remarks and discussion, if the Examiner still maintains that Applicants' claims are anticipated by Koblenz' 829, meaning that Koblenz' 829 teaches a bank number indicating the number of times the first or head pointer has wrapped around the last or tail pointer, Applicants request that the Examiner point it out with particularity.

Column 11, lines 10-55 of Koblenz '829 discuss Figure 4 which is a block diagram of the bucket array and warehouse data structures. Each bin of the bucket array has a data structure having the variables of Table 1. The variables include a

bin\_tail pointer to the tail portion of the six list data structure and a bin\_circle pointer to the circle portion of the same six list data structure. A bin\_highwater variable indicates the maximum number of lockers that have been simultaneously allocated for the warehouses in the bin; and the bin\_highwater variable is incremented and decremented when one of the lockers is freed in the circle portion up to the highwater mark at which time a new warehouse is allocated. The variables, however, do not include the number of times that bin\_circle has wrapped around the bin\_tail!

Koblenz '829 at column 14, line 54 through column 15, line 25 discusses Figure 11 which is a flow chart of how to allocate a memory component. First, the routine returns a pointer to a block of memory and then determines if this block of memory is sufficient to satisfy the request. If so, then the routine calculates the virtual bin number using a floating point technique and maps the virtual bin number to the actual bin number. The routine then retrieves a pointer and subtracts [adds -1] from the number of free lockers. If there are not a sufficient number of free lockers available in the bin, then a new warehouse of 64 lockers must be allocated. In any event, once a new warehouse is allocated, a warehouse header to a circle portion of the six list is added. If there are a sufficient number of free lockers, then the routine gets a locker from the existing warehouse. Again, the increment/decrement value refers only to the number of free lockers [bin\_netfree] in a warehouse. The circle pointer never passes the tail pointer and this is not recorded in a bank number, as applicants claim.

Koblenz '892 at column 16, line 37 through column 17, line 6 discusses Figure 14 which is a flow chart of how an available warehouse header is retrieved from the tail portion of the six list. A description of the reallocation of next and previous pointers follows as the available warehouse header is retrieved from the tail portion and is linked into the circle portion. The routine adjusts the number of available lockers, and a new thread will take note that a warehouse header is available for

use. Again, Attorney for Applicants have not been able to discern that Koblenz '829 teaches that a bank number keeps track of the number of times a circle pointer passes the tail pointer.

Respectfully, Table 1, Table 2, the description of the six list data structure with warehouse headers being allocated or freed simply do not teach or suggest the claimed "bank number to indicate how many times the head pointer/first entry has wrapped around the tail pointer/last entry in a resource queue" as Applicants claim in independent claims 1 and 7. Applicants respectfully request the Examiner enter the amendments and remove the rejection under 35 U.S.C. §102(e) under Koblenz' 829.

*The Rejection of claim 5 Under 35 U.S.C. §103(a) over Koblenz '829 in view of Lee '271 and the Rejection of claims 6, 10-13, and 15 Under 35 U.S.C. §103(a) over Lee '271 in view of Koblenz '829.*

The Examiner rejected claim 5 under 35 U.S.C. §103(a) under a combination of Koblenz '829 and Lee '271, using Koblenz '829 as above. The Examiner admits that Koblenz '829 does not teach an out-of-order computer processor having a resource queue comprising a load reorder queue and/or a store reorder queue and/or a global completion table and/or a branch information queue. The Examiner asserts that those elements are taught by Lee '271, and that a person of ordinary skill in the art would incorporate the out-of-order methods of Lee '271 in the device of Koblenz '829 to improve processor speed and efficiency.

The Examiner also rejected claims 6, 10-13, and 15 under 35 U.S.C. §103(a) over Lee '271 in view of Koblenz '829. The Examiner states here that Lee '271 teaches an out-of-order processor but admits that Lee '271 does not, but Koblenz '829 does, teach a resource queue having a first entry of one thread capable of wrapping around a last entry of the same thread and then using a bank number to indicate how many times the head pointer wraps around the tail pointer. The

Examiner reasons that a person of ordinary skill in the art would incorporate the circular queue of Koblenz '829 into the device of Lee '271 to improve processor efficiency.

Applicants traverse the rejection of claims 5, 6, 10-13, and 15 under the alleged combination of Lee '271 and Koblenz '829 on two grounds: first, neither Koblenz '829 nor Lee '271 teach a bank number keeping track of how many times a first pointer has passed a last pointer in a resource queue to maintain in-order processing in an out-of-order processor; second, Koblenz '829 does not suggest that its six list can be used in an out-of-order processor and, absent the Examiner's suggestion, there is no suggestion in either reference for the alleged combination. In fact, Koblenz '829 teaches against out-of-order processing.

With respect to the first traversal of the rejection of the claims under 35 U.S.C. §103(a), Applicants respectfully maintain that Koblenz '829 does not teach a resource queue in a processor wherein the resource queue has bank number indicating the number of times a circle pointer of one thread has passed a tail pointer of the same thread. Respectfully, Applicants assert the arguments above as applied to the anticipation rejection under 35 U.S.C. §102(e) by Koblenz '829. Koblenz '829 teaches only that a circle pointer/first entry points to the next warehouse header in memory which points to the next warehouse header in memory, etc. and that a tail pointer/last entry points to the previous warehouse header in memory which points to the previous warehouse header in memory. Koblenz '829 does not even teach or suggest that its circular six list can be used as a resource queue in a processor; it is used only in the allocation of small memory. The Examiner admits that Lee '271 does not teach the resource queue as claimed.

With respect to the second argument, Applicants assert that Koblenz '829 teaches away from the alleged combination of using a circular six-list for small memory allocation with an out-of-order processor. Koblenz '829 teaches fine-grained multithreading which interleaves or switches threads on a cycle-by-cycle

basis, i.e., “one thread of execution out of 128 streams supported by a processor executes on every clock cycle ... Thus, a new instruction for a different stream may be issued in each time period without interfering with other instructions that are in the pipeline. When an instruction finishes, the stream to which it belongs becomes ready to execute the next instruction.” Koblenz ‘829 at column 1, lines 28 through 41. An out-of-order processor, on the other hand, is one in which an instruction is allowed to complete before all instructions ahead of it has been completed (Applicants’ specification at page 5, lines 14-15). Koblenz ‘829 does not suggest or even hint of an out-of-order processor and doesn’t even mention the pipeline stages in the processor. Koblenz ‘829 teaches away from a shared resource queue in its pipeline: at column 1, lines 44-48, the reference states that it has a stream status word, 32 general registers, and eight target registers for each of the 128 threads - that’s a lot of registers, a lot of silicon, i.e., Koblenz ‘829 teach separate resources for each thread in the processor pipeline and certainly does not suggest a shared resource queues and does not suggest an out-of-order processor.

Applicants request the Examiner withdraw the rejection of claims 5, 6, 10-13, and 15 under 35 U.S.C. §103(a) under the alleged combination of Koblenz ‘829 and Lee ‘271.

### Conclusion

Applicants maintain that Koblenz ‘829 does not teach that a bank number keeps track of the number of times a head pointer passes a tail pointer in a resource queue shared amongst simultaneous threads in a processor. Koblenz ‘829, moreover, teaches away from out-of-order processing using shared queues within the processor pipeline. Lee ‘271, while teaching an out-of-order processor, does not suggest a resource queue having resources for more than one thread in which a bank number indicates the number of times a head pointer of a thread can wrap

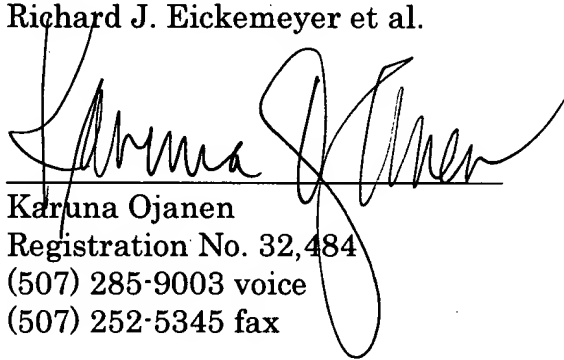
around the tail pointer of the same thread. The Examiner is not free to suggest a combination where the alleged reference teach away from the combination.

In view of the remarks above, Attorney for Applicants maintains the patentability of the claimed invention herein. Multithreaded processing having hardware resources shared by more than one thread and keeping track of the order of the resources allocated to each thread by allowing a head pointer to surpass a tail pointer and then keeping track of the number of passes in an out-of-order processor is not trivial, is not a mere combination nor an obvious modification of the two references. Applicants thus respectfully request the Examiner to enter the amendments because they place the claims in condition for allowance and/or in better condition for appeal. The Examiner is further invited to telephone the Attorney listed below if he thinks it would expedite the prosecution and the issuance of the patent.

Respectfully submitted,  
Richard J. Eickemeyer et al.

Date: 12 March 2005

By



Karuna Ojanen  
Registration No. 32,484  
(507) 285-9003 voice  
(507) 252-5345 fax

IBM Corporation  
Intellectual Property Law Dept. 917  
3605 Highway 52 North  
Rochester, MN 55901-7829